What techniques and deployment modes will tend to improve trust in automatically improved code and how do we surface and evaluate these “trust issues” as a research questions?

Stefan Forstenlechner
Natural Computing Research & Applications Group
University College Dublin
Psychology

• How did people start to trust compilers?

• What is the largest code change developers accept (from humans/machines)?

• Humans might only use what they know.
Psychology

• Gradually introduce GI
  • Learn in the background
  • Hints/Suggestions
  • Feedback
  • Small changes
  • Slowly increase size of changes

• Can "over-trusting" become an issue?
Double deployment

• Can you run human and GI improved code in parallel?

• Use one to check the other

• In case of success, slowly replace human code
Build trust

• Usability
  • GI should not be a hassle
  • GI should add benefit

• Success stories
  • Convince everyone from the developer to the upper management

• Open source vs closed source
Where is GI more trustworthy than a human?

• Response time

• Iterative process
  • Can show intermediate results

• People can be protective of their code. GI does not care.

• GI does not have human weaknesses
Why GI is better ;) 

• Humans join/leave teams
• GI can learn from mistakes
• GI can help train new developers
Refactoring

• Developers do not like refactoring

• GI can learn or be told the rules

• GI can help guide or even apply rules
Getting feedback from a human vs machine

• Machine might not be as trusted (at first)

• Especially for new developers feedback from machines might not be as embarrassing
Acknowledgements

```
$ apt-get moo

(____)
 ( 00 )
/------\/
 /   ||
 *   \--\--
~~   ~~

"Have you mooed today?"
```